Switching Circuits & Logic Design

Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University

Fall 2013

§7 Multi-Level Gate Circuits

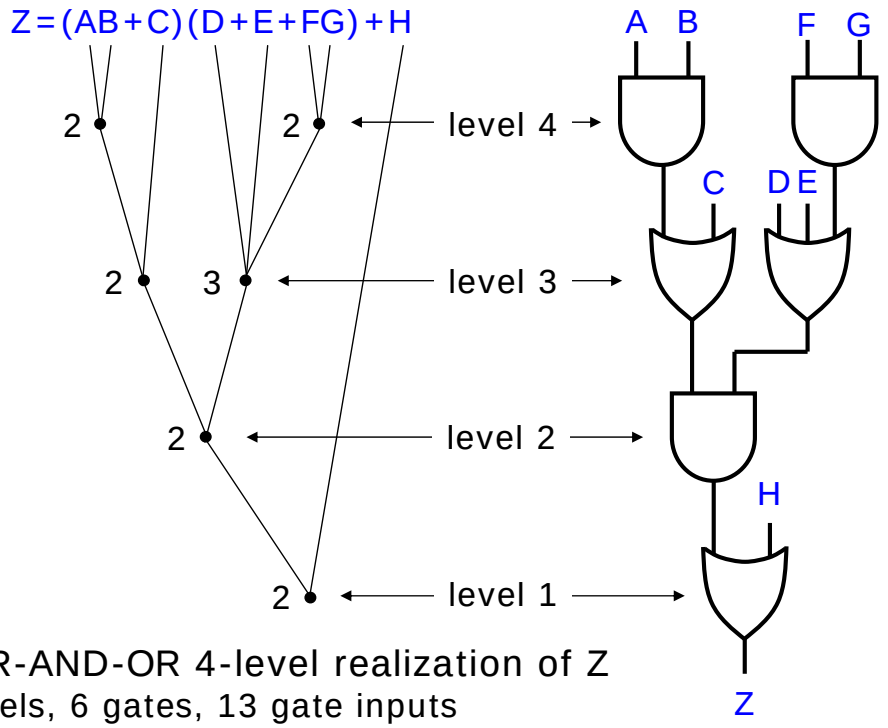
Outline

- Multi-level gate circuits
- NAND and NOR gates
- Design of two-level circuits using NAND and NOR gates
- Design of multi-level NAND- and NOR-gate circuits
- Circuit conversion using alternative gate symbols
- Design of two-level, multiple-output circuits
- Multiple-output NAND and NOR circuits

Multi-Level Gate Circuits

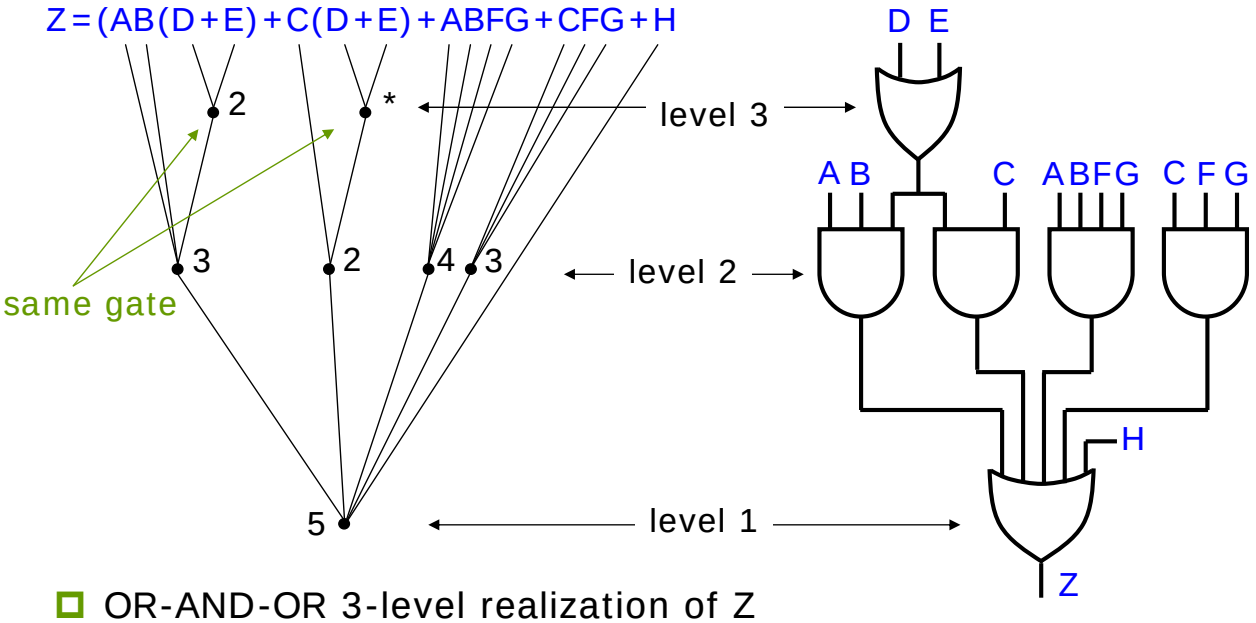
- The number of **levels** of gates
  - The maximum number of gates cascaded in series between a circuit input and output
    - Inverters which are connected directly to input variables do not count (assume variables and their complements are available as circuit inputs)
  - SOP (POS) correspond to AND-OR (OR-AND) two-level gate circuits
    - AND-OR
      - 2-level circuit composed of a level of AND gates followed by an OR gate at the output
    - OR-AND
      - 2-level circuit composed of a level of OR gates followed by an AND gate at the output
    - OR-AND-OR
      - 3-level circuit composed of a level of OR gates followed by a level of AND gates followed by an OR gate at the output
    - Circuit of AND and OR gates
      - No particular order of the gates

Multi-Level Gate Circuits



- AND-OR-AND-OR 4-level realization of Z
  - 4 levels, 6 gates, 13 gate inputs

Multi-Level Gate Circuits



- OR-AND-OR 3-level realization of Z
  - Partially multiplying out Z = (AB+C)[(D+E)+FG]+H
  - 3 levels, 6 gates, 19 gate inputs

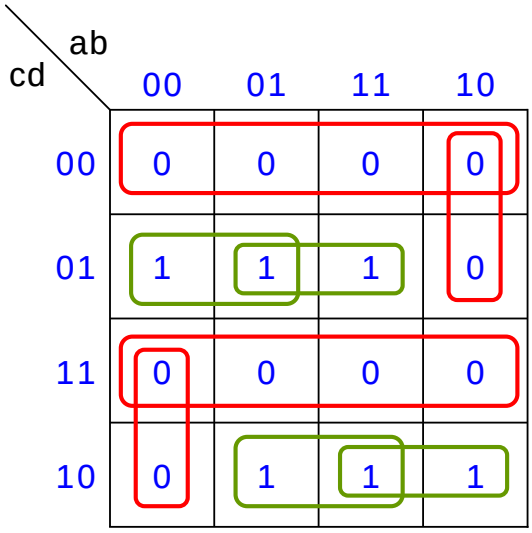
Multi-Level Gate Circuits

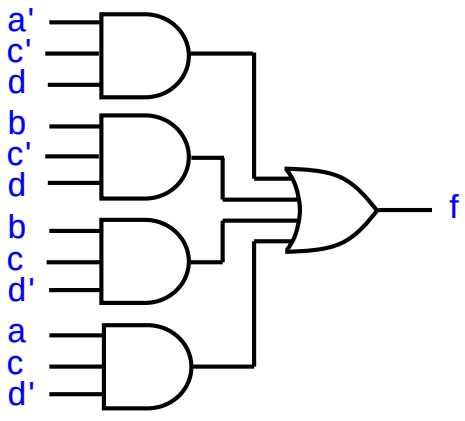
- Drawing the tree diagram of an expression helps determine the realization costs
  - #level → circuit delay
  - #gates, #gate inputs → circuit area
- Different expressions of a Boolean function provide different tradeoffs between delay and area

Multi-Level Gate Circuits

- Find a circuit of AND and OR gates realizing f(a,b,c,d) = ∑m(1,5,6,10,13,14)

f = a'c'd+bc'd+bcd'+acd'



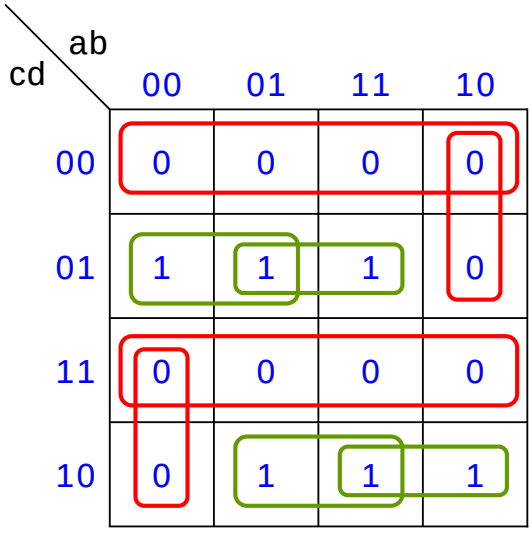


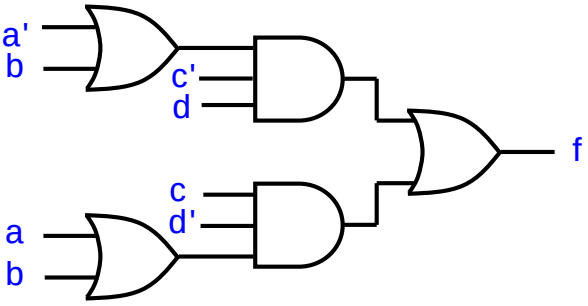
2 levels, 5 gates, 16 gate inputs

Multi-Level Gate Circuits

- Example (cont'd)

f = a'c'd+bc'd+bcd'+acd'
  = c'd(a'+b)+cd'(a+b)



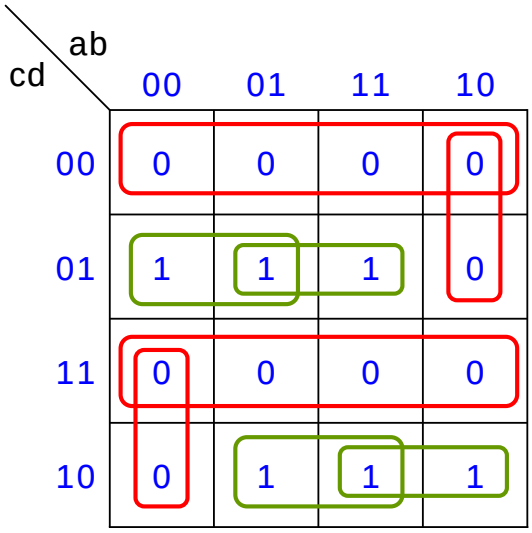


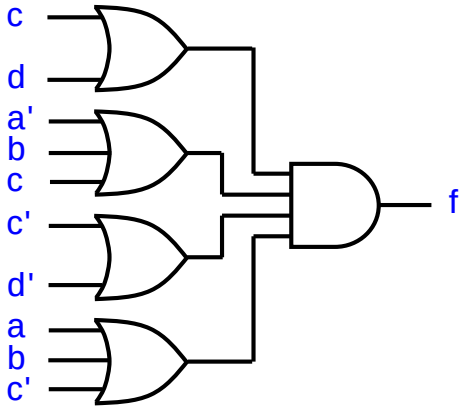
3 levels, 5 gates, 12 gate inputs

Multi-Level Gate Circuits

- Example (cont'd)

f' = c'd'+ab'c'+cd+a'b'c
f = (c+d)(a'+b+c)(c'+d')(a+b+c')





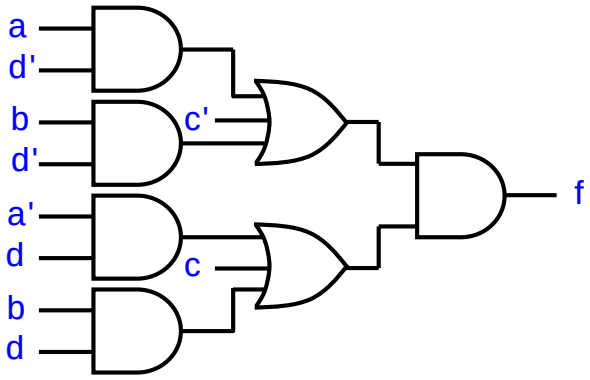
2 levels, 5 gates, 14 gate inputs

Multi-Level Gate Circuits

- Example (cont'd)

f = (c+d)(a'+b+c)(c'+d')(a+b+c')
= (c+a'd+bd)(c'+ad'+bd')

cd \ ab	00	01	11	10
00	0	0	0	0
01	1	1	1	0
11	0	0	0	0
10	0	1	1	1



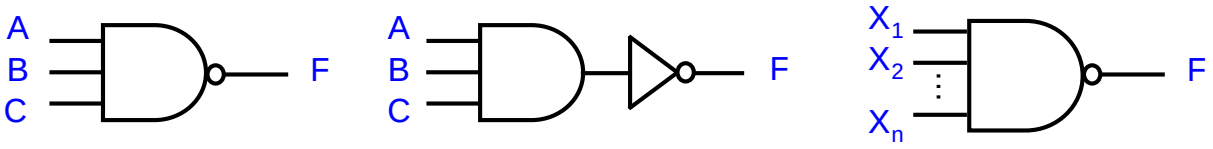
3 levels, 7 gates, 16 gate inputs

Multi-Level Gate Circuits

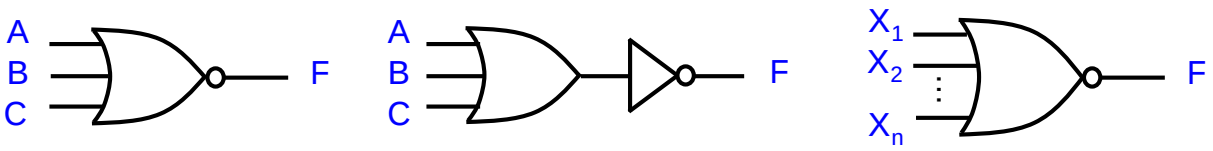
- To be sure of obtaining a minimum solution, we have to find both the circuit with the AND-gate output and the one with the OR-gate output
- If the expression for f' has n levels with an AND-gate (OR-gate) output, its complement is an n-level expression for f with an OR-gate (AND-gate) output

NAND and NOR Gates

- NAND
  - AND-NOT gate
  - 3-input NAND: F = (ABC)' = A'+B'+C'
  - n-input NAND: F = $(X_1X_2...X_n)' = X_1'+X_2'+...+X_n'$

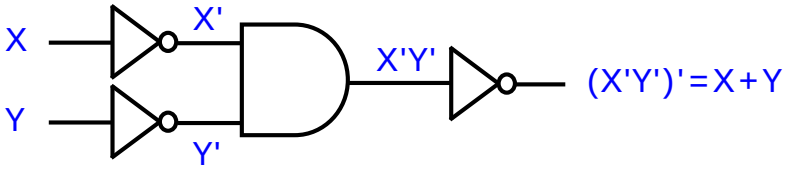


- NOR
  - OR-NOT gate
  - 3-input NOR: F = (A+B+C)' = A'B'C'
  - n-input NOR: F = $(X_1+X_2+...+X_n)' = X_1'X_2'...X_n'$



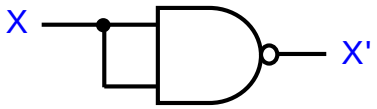
NAND and NOR Gates

- A set of logic operations is **functionally complete** if any Boolean function can be expressed in terms of the set of operations
  - {AND, OR, NOT} is functionally complete
    - Any Boolean function can be expressed in SOP form, which uses only the AND, OR, NOT operations
  - Any set of logic operations that can realize AND, OR, NOT is also functionally complete
    - {AND, NOT} is functionally complete since OR can be realized using AND and NOT as shown below

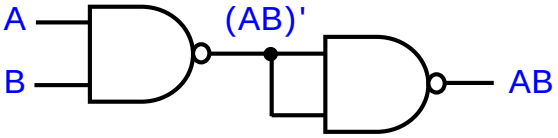


NAND and NOR Gates

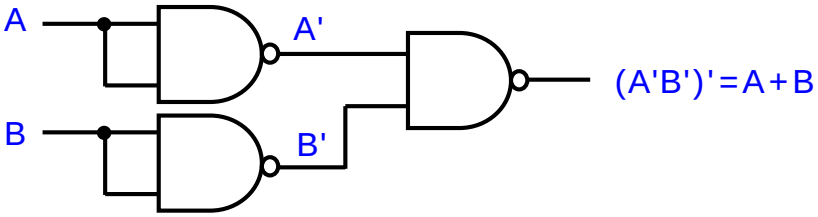
- {NAND} is functionally complete
  - NOT: (X·X)' = X'



  - AND: ((A·B)')' = A·B



  - OR: (A'·B')' = A+B



NAND and NOR Gates

- How to show whether or not a set of logic operations is functionally complete?
  1. Write out a minimum SOP expression for the function realized by each gate
  2. If no complement appears in any of these expressions, then NOT cannot be realized
  3. Otherwise, NOT can be realized by an appropriate choice of inputs to the corresponding gate (assume 0 and 1 are available as gate inputs)
  4. Try to realize AND or OR (now with NOT available)

NAND and NOR Gates

- Exercises
  - Show that the sets {NOR} and {OR, NOT} are functionally complete
  - Is the majority gate *major* functionally complete?
    - *major*(A,B,C) = 1 iff at least two of A, B, C are 1
  - Is the minority gate *minor* functionally complete?
    - *minor*(A,B,C) = 1 iff at most one of A, B, C is 1
  - Is {→} functionally complete?
    - A→B is true iff A is false (0), or both A and B are true (1)
    - Does the assumption "0 and 1 are available as gate inputs" make a difference?

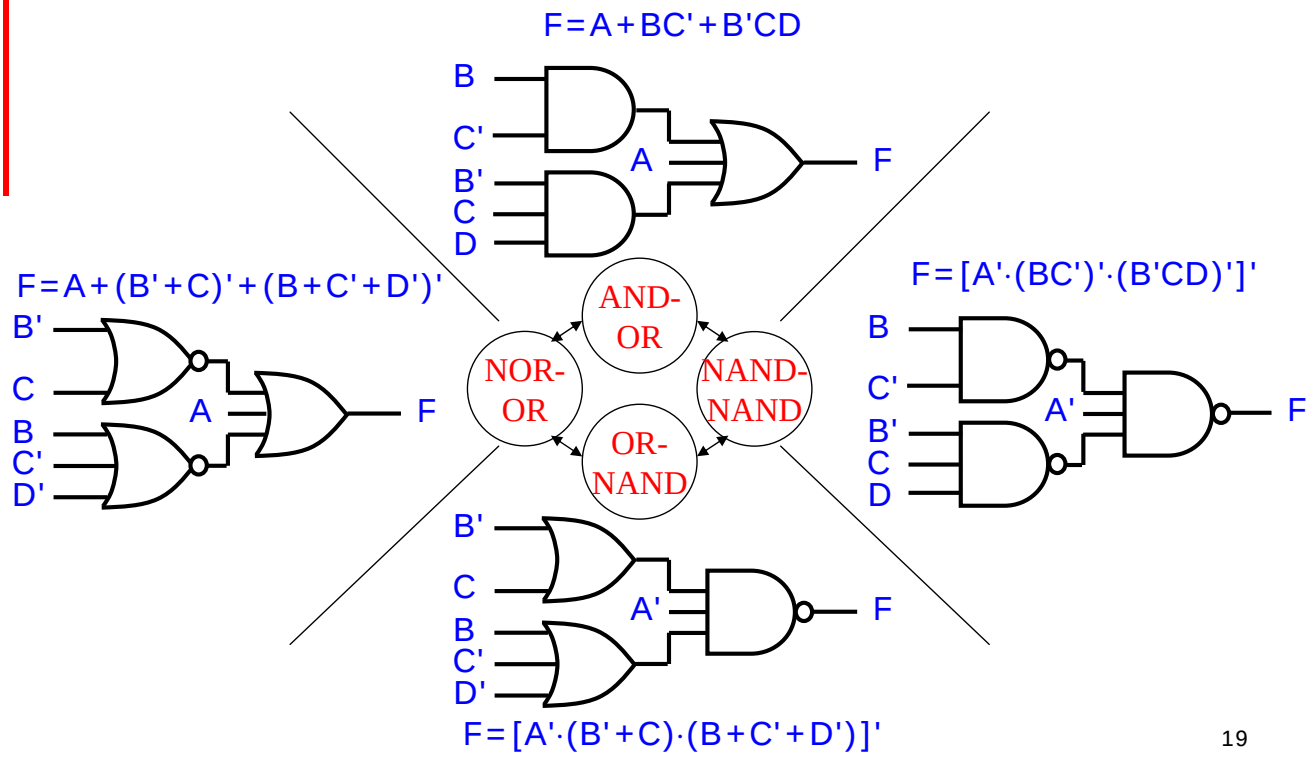
Two-Level Circuit Design Using NAND and NOR Gates

- A 2-level circuit composed of AND, OR gates can be converted to a circuit composed of NAND, NOR gates
  - By using F = (F')' and DeMorgan's laws

Example 1

F = A+BC'+B'CD (AND-OR)
= [(A+BC'+B'CD)']'
= [A'(BC')'(B'CD)']' (NAND-NAND)
= [A'(B'+C)(B+C'+D')]' (OR-NAND)
= A+(B'+C)'+(B+C'+D')' (NOR-OR)

Two-Level Circuit Design Using NAND and NOR Gates

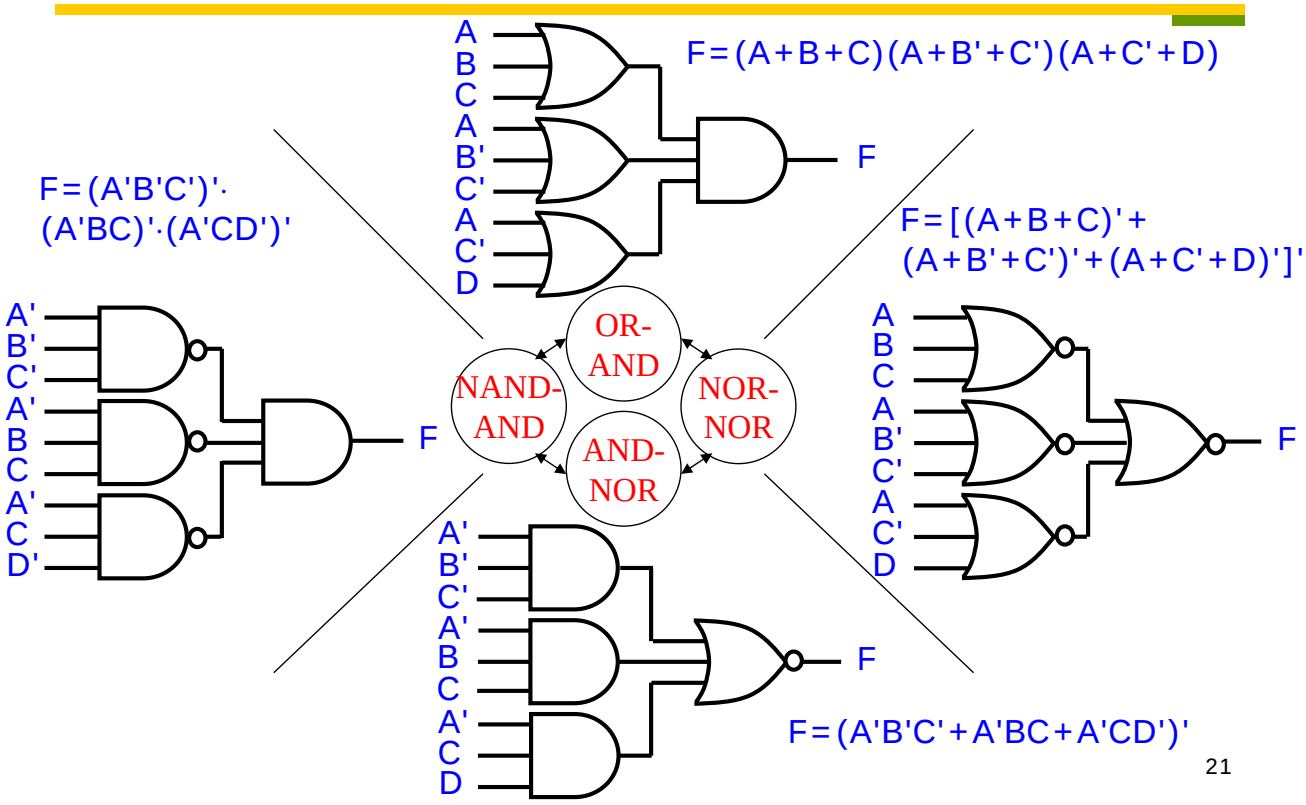


Two-Level Circuit Design Using NAND and NOR Gates

Example 2

F = (A+B+C)(A+B'+C')(A+C'+D) (OR-AND)
= {[(A+B+C)(A+B'+C')(A+C'+D)]'}'
= [(A+B+C)'+(A+B'+C')'+(A+C'+D)']' (NOR-NOR)
= [(A'B'C')+(A'BC)+(A'CD')]' (AND-NOR)
= (A'B'C')'(A'BC)'(A'CD')' (NAND-AND)

Two-Level Circuit Design Using NAND and NOR Gates

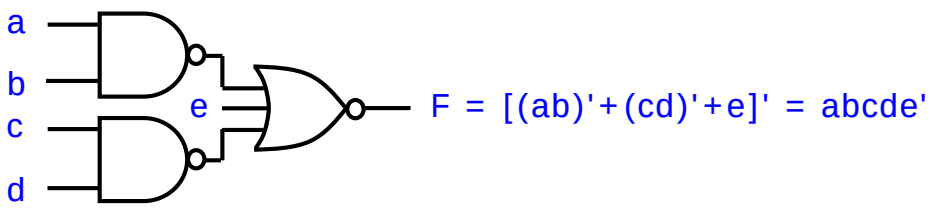


Two-Level Circuit Design Using NAND and NOR Gates

- Among the 16 two-level forms:
  - The following 8 are generic (can realize all switching functions):
    AND-OR, AND-NOR, OR-AND, OR-NAND, NAND-AND, NAND-NAND, NOR-OR, NOR-NOR
  - The following 8 are degenerate (cannot realize all functions):
    AND-AND, AND-NAND, OR-OR, OR-NOR, NAND-OR, NAND-NOR, NOR-AND, NOR-NAND

E.g.,

NAND-NOR form can realize only a product of literals and not a sum of products

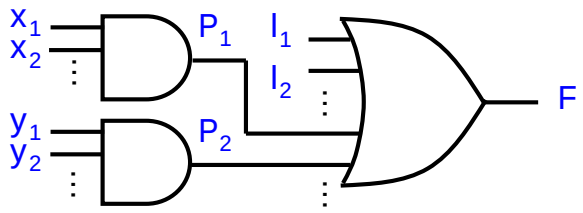


Two-Level Circuit Design Using NAND and NOR Gates

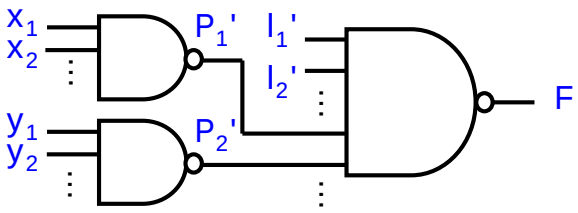
- NAND-NAND and NOR-NOR are the most widely used forms in integrated circuits

- Procedure for minimum NAND-NAND (NOR-NOR) implementation
  1. Find a minimum SOP expression of F
  2. Draw the corresponding two-level AND-OR (OR-AND) circuit
  3. Replace all gates with NAND (NOR) gates with interconnections unchanged. Complement the single-literal inputs of the output gate

$F = I_1 + I_2 + \cdots + P_1 + P_2 + \cdots$



$F = (I_1' I_2' \cdots P_1' P_2' \cdots)'$



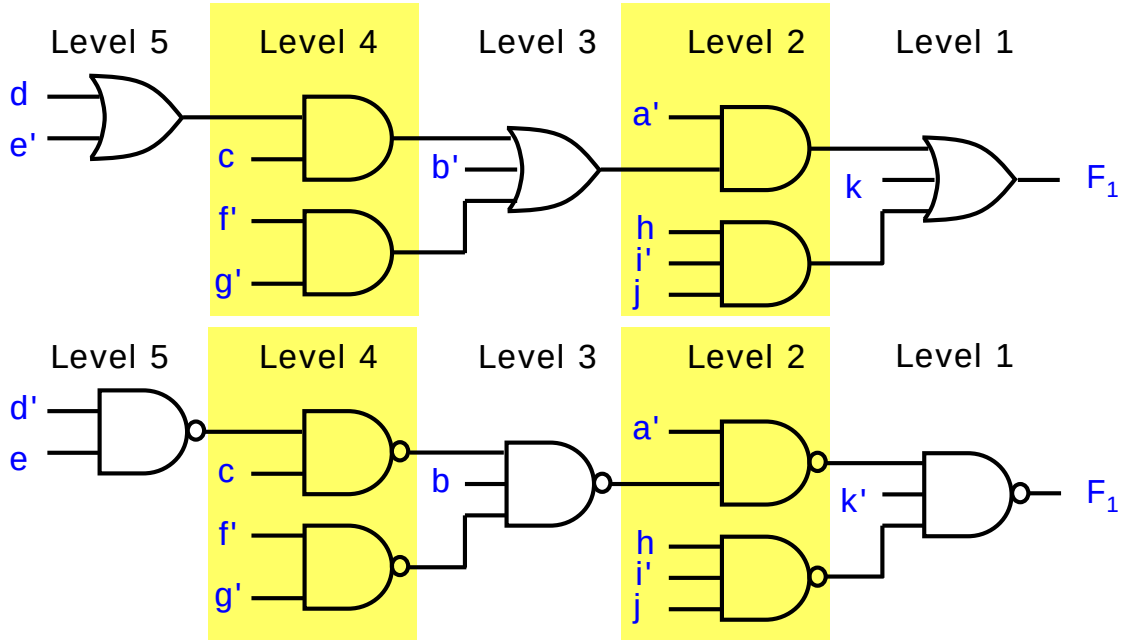
Design of Multi-Level NAND- and NOR-Gate Circuits

- Procedure for designing multi-level NAND-gate circuits
  1. Simplify F
  2. Design a multi-level circuit of AND and OR gates with output gate being OR
     - AND-gate (OR-gate) outputs cannot be used as AND-gate (OR-gate) inputs
  3. Number the levels starting with the output gate as level 1. Replace all gates with NAND gates, leaving interconnections unchanged. Invert any literals which appear as inputs to levels 1, 3, 5, ...

Design of Multi-Level NAND- and NOR-Gate Circuits

Example

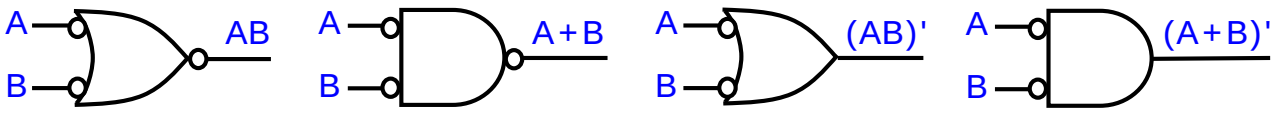
$F_1 = a'[b'+c(d+e')+f'g']+hi'j+k$



Circuit Conversion Using Alternative Gate Symbols

- Alternative gate symbols
  - Useful for circuit analysis and design

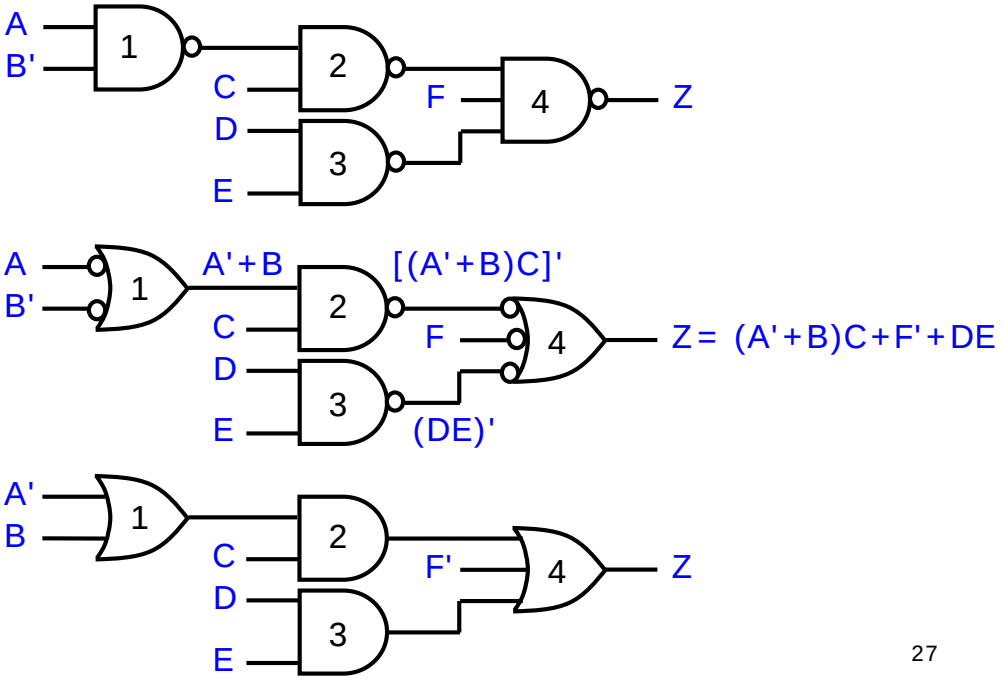
NOT



AND OR NAND NOR

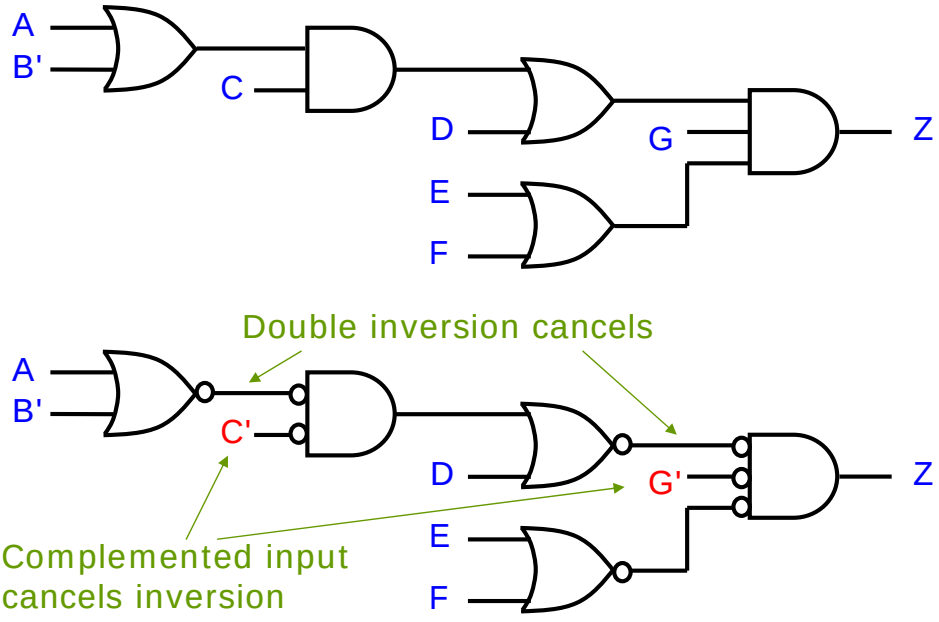
Circuit Conversion Using Alternative Gate Symbols

- NAND gate circuit conversion



Circuit Conversion Using Alternative Gate Symbols

- Conversion to NOR gates

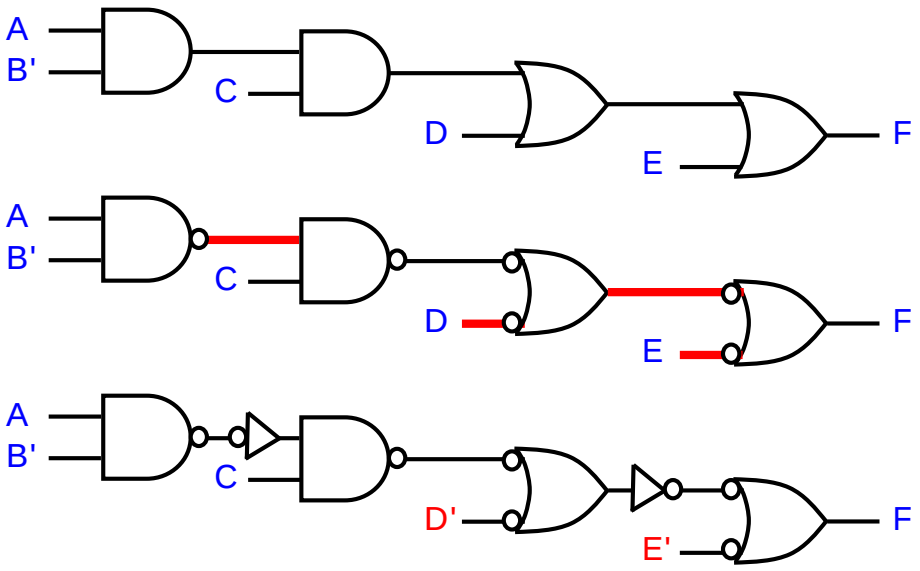


Circuit Conversion Using Alternative Gate Symbols

- A circuit composed of AND and OR gates can be converted to a circuit composed of NAND and NOR gates, and vice versa
  - By properly adding inverters, removing canceling inverter pairs, and/or negating inputs, we can change a gate type to a desired one

Circuit Conversion Using Alternative Gate Symbols

- Conversion to NAND gates (even if AND and OR gates do not alternate)



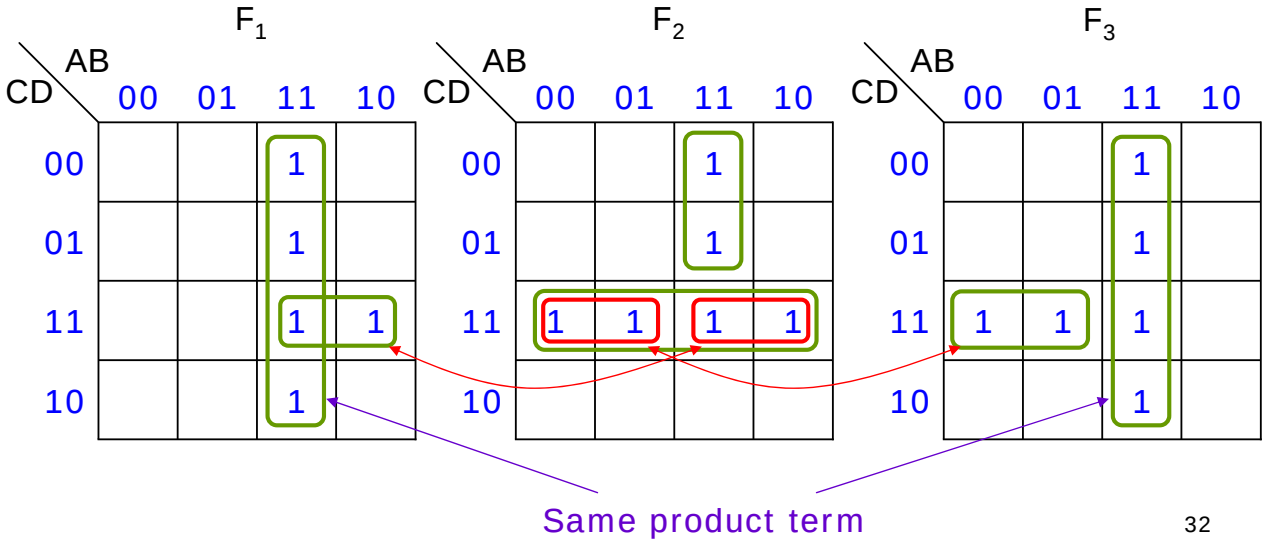
Two-Level, Multiple-Output Circuit Design

- In circuit design, often we need several Boolean functions rather than one
  - Although every function can be implemented separately, recognizing common gates among these functions can achieve logic sharing and thus reduce area
  - When designing multiple-output circuits, we should try to first minimize the total number of gates required and then minimize gate inputs

Two-Level, Multiple-Output Circuit Design

Example

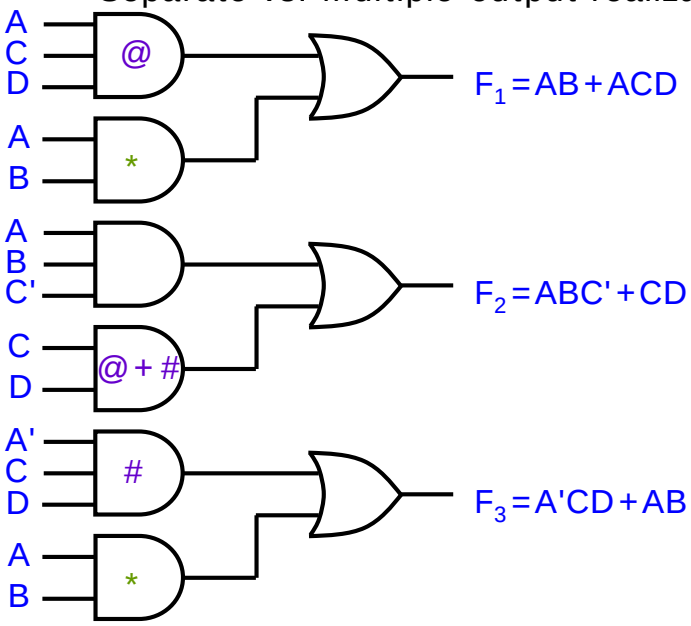
$F_1(A,B,C,D) = \sum m(11,12,13,14,15)$
$F_2(A,B,C,D) = \sum m(3,7,11,12,13,15)$
$F_3(A,B,C,D) = \sum m(3,7,12,13,14,15)$



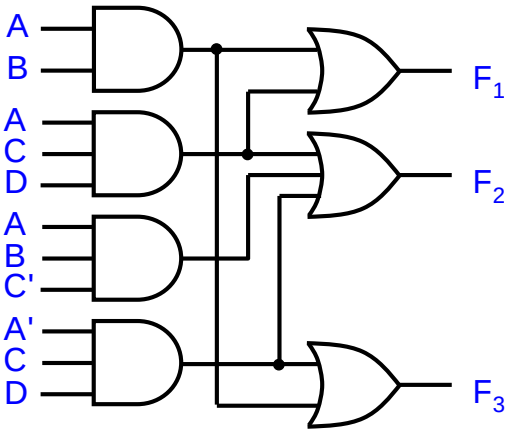
Two-Level, Multiple-Output Circuit Design

Example (cont'd)

Separate vs. multiple-output realization



9 gates, 21 gate inputs



7 gates, 18 gate inputs

Note that F_2 in the multiple-output realization is not a minimum SOP

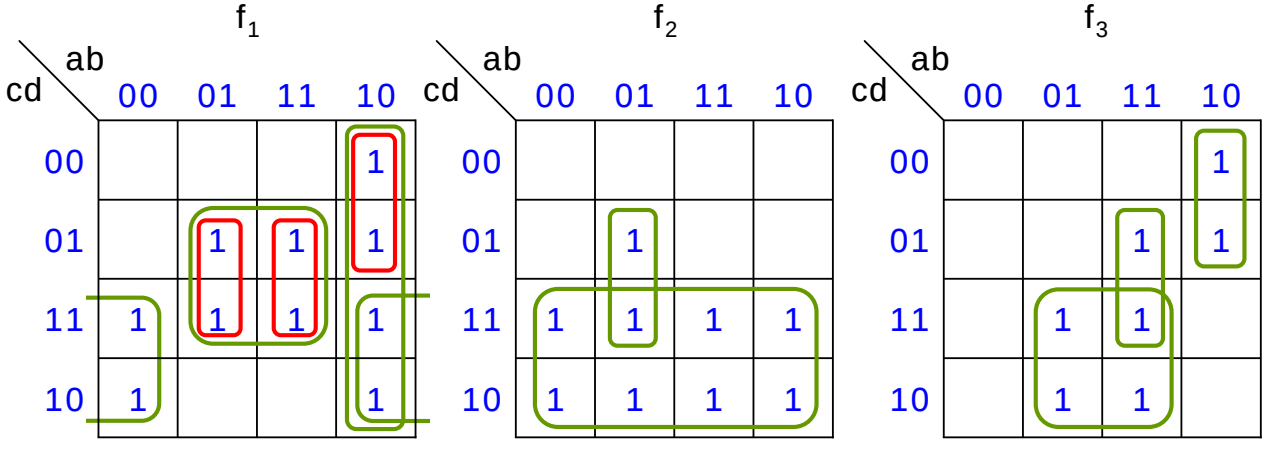
Two-Level, Multiple-Output Circuit Design

Example

$f_1(a,b,c,d) = \sum m(2,3,5,7,8,9,10,11,13,15)$

$f_2(a,b,c,d) = \sum m(2,3,5,6,7,10,11,14,15)$

$f_3(a,b,c,d) = \sum m(6,7,8,9,13,14,15)$



Two-Level, Multiple-Output Circuit Design

Example (cont'd)

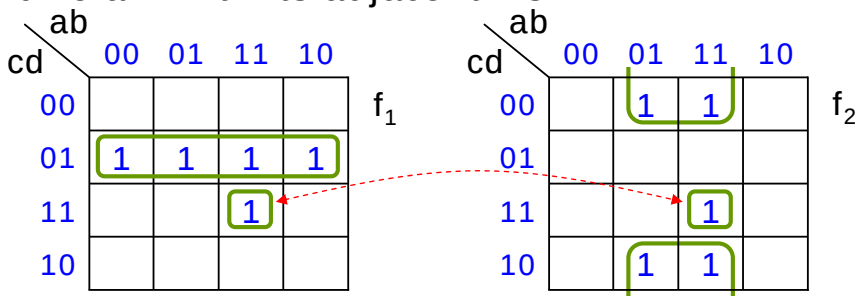
Separate realization	Multi-output realization
$f_1 = bd+b'c+ab'$	$f_1 = a'bd+abd+ab'c'+b'c$
$f_2 = c+a'bd$	$f_2 = c+a'bd$
$f_3 = bc+ab'c'+abd$	$f_3 = bc+ab'c'+abd$
10 gates, 25 gate inputs	8 gates, 22 gate inputs

Two-Level, Multiple-Output Circuit Design

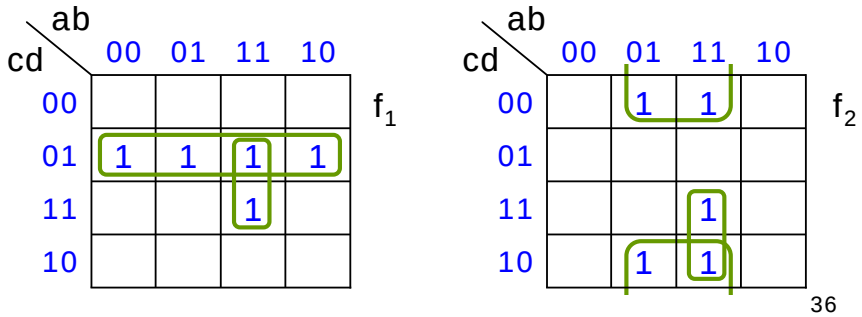
Example

- When designing multiple-output circuits, it is sometimes best not to combine a 1 with its adjacent 1's

Best solution:



Solution requires an extra gate:

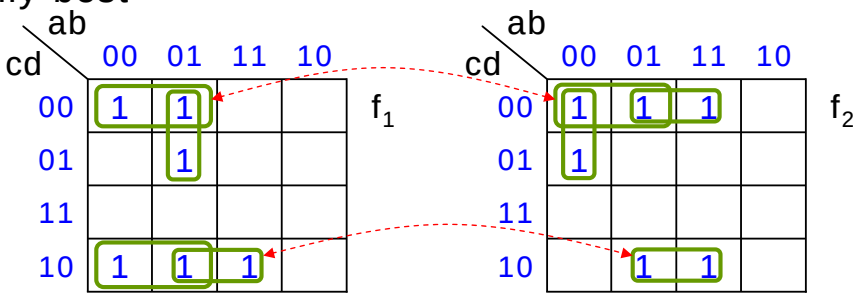


Two-Level, Multiple-Output Circuit Design

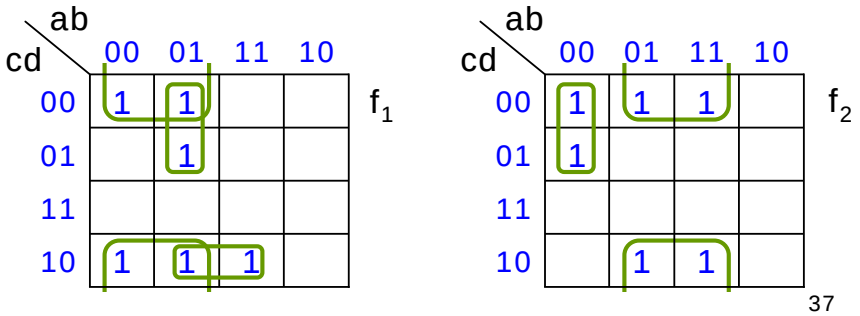
Example

- The solution with the maximum number of common terms is not necessarily best

Solution with maximum # of common terms: (8 gates, 26 inputs)



Best solution (no common terms): 7 gates, 18 inputs



Two-Level, Multiple-Output Circuit Design

- Procedure of SOP minimization applies for multiple-output realization with slight modification of the determination of essential prime implicants (EPIs)
  - We find prime implicants that are essential to one of the functions and to the multiple-output realization
    - Some of the prime implicants essential to an individual function may not be essential to the multiple-output realization
      - Recall an EPI is a prime implicant that covers some minterm m_i that is not covered by any other prime implicant
      - The minterm m_i may appear and be covered by a (different ?) prime implicant in another function

E.g.,

Slide 34:

bd is essential to f_1, but not for the multiple-output realization

Slide 36:

c'd is essential to f_1 for the multiple-output realization

abd is essential to f_1, but not for the multiple-output realization

Slide 37:

a'd', a'bc' are essential to f_1 for the multiple-output realization

bd' is essential to f_2 for the multiple-output realization

Multiple-Output NAND and NOR Circuits

- Procedure for designing single-output, multi-level NAND- and NOR-gate circuits applies to multiple-output circuits
  - If all of the output gates are OR (AND), direct conversion to a NAND-gate (NOR-gate) circuit is possible

Multiple-Output NAND and NOR Circuits

- Multi-level circuit conversion to NOR gates